رهیافت توماس موازی در دینامیک سیالات محاسباتی به‌کمک پردازنده‌های گرافیکی – جریان درون حفره*

(یادداشت پژوهشی)

پوریا اکبرزاده[(۱)] حسین محمودی داریان[(۲)] محسن نظری[(۳)] میلاد سوری[(۴)]

چکیده در این مقاله سه الگوریتم کاهش‌متناوب، کاهش‌متناوب موازی و رهیافت توماس موازی برای حل دستگاه معادلات سه‌قطری به‌کمک پردازنده‌های گرافیکی معرفی و اثر دسترسی هم‌مکان و غیرهم‌مکان به حافظهٔ سراسری مورد بحث قرار گرفته‌است. برای ارزیابی توانایی این الگوریتم‌ها، نتایج شبیه‌سازی جریان درون حفره (یک مورد مطالعاتی) با نتایج الگوریتم توماس‌کلاسیک اجراشده روی پردازندهٔ مرکزی مقایسه شده‌است. بیشینه افزایش سرعت مشاهده‌شده در سه الگوریتم مذکور (پردازنده‌گرافیکی) در برابر الگوریتم توماس کلاسیک (پردازندهٔ مرکزی) به‌ترتیب حدود ۴/۴، ۵/۲ و ۳۸/۴۵ می‌باشد. هم‌چنین نشان داده شده است که با دسترسی هم‌مکان، افزایش سرعت حدوداً دوبرابری برای پردازندهٔ گرافیکی حاصل می‌شود.

واژه‌های کلیدی رهیافت توماس موازی; پردازش موازی; دستگاه معادلات سه‌قطری; پردازندهٔ گرافیکی; الگوریتم کاهش متناوب.

Parallel Thomas Approach in Computational Fluid Dynamics with GPUs– Lid-driven Cavity

P. Akbarzadeh H. Mahmoodi Darian M. Nazari M. Souri

Abstract *In this paper three algorithms of Cyclic-Reduction, Parallel-Cyclic-Reduction and Parallel-Thomas are introduced to solve the Tridiagonal system of equations using GPUs and the effect of coalesced-memory-access and uncoalesced-memory-access to global memory are studied. To assess the ability of these algorithms, as a case-study the simulation of the lid-driven cavity flow have been compared to the results of Runtimes and physical parameters of the classical Thomas algorithm, executed on CPU. The maximum speed-up of these algorithms against CPU runtime is about 4.4x, 5.2x and 38.5x, respectively. Also, approximately a 2x speed-up achieved in coalesced-memory access on GPU.*

Key Words Parallel Thomas approach; Parallel Processing; Tridiagonal system of equations; Graphic Processor; Cyclic Reduction algorithm

---

★تاریخ دریافت مقاله ۹۴/۴/۲۳ و تاریخ پذیرش آن ۹۴/۹/۱۸ می‌باشد. DOI: 10.22067/fum-mech.v28i2.48399

(۱) نویسندهٔ مسئول: دانشیار، مهندسی مکانیک، دانشگاه شاهرود. p.akbarzadeh@shahroodut.ac.ir

(۲) استادیار، مهندسی مکانیک، دانشگاه تهران.

(۳) دانشیار، مهندسی مکانیک، دانشگاه شاهرود.

(۴) دانشجوی کارشناسی ارشد، مهندسی مکانیک، دانشگاه شاهرود.

مقدمه

در سال‌های اخیر باتوجه به نیاز جوامع علمی به بررسی مسائل مختلف با پیچیدگی بیشتر، نیاز به کامپیوترهایی با سرعت و حافظهٔ بیشتر احساس می‌شد، بنابراین محققان به استفاده از پردازنده‌های مرکزی (Central Processor Unit, CPU) با سرعت بیشتر روی آوردند. در کنار این رویکرد، امروزه بهره‌گیری از پردازنده‌های گرافیکی (Graphic Processor Units, GPU) نیز، به‌دلیل عملکرد بسیار بالا و پهنای باند(Memory Band Width) حافظه زیاد، جایگاه ویژه‌ای را در شبیه‌سازی پدیده‌ها و محاسبات علمی به خود اختصاص داده‌اند. ساختار معماری پردازنده‌های گرافیکی به‌گونه‌ای است که می‌تواند در زمانی کوتاه یک دستورالعمل محاسباتی را روی تعداد زیادی داده به‌صورت موازی و هم‌زمان اجرا نماید. روند روبه رشد استفاده از پردازنده‌های گرافیکی به‌عنوان شتاب‌دهندهٔ محاسبات به‌گونه‌ای است که ابرکامپیوترهای برتر حال حاضر دنیا از قدرت پردازنده‌های گرافیکی بهره می‌برند [1]. در بسیاری از مسائل مهندسی، گسسته‌سازی معادلات حاکم منجر به تشکیل یک دستگاه معادلات سه‌قطری می‌شود که حل سریع این دستگاه معادلات یک موضوع چالش‌برانگیز در دینامیک سیالات محاسباتی محسوب می‌گردد. یکی از روش‌های قدیمی اما رایج و پرسرعت برای حل دستگاه معادلات سه‌قطری، الگوریتم توماس است که برمبنای روش حذفی گوس می‌باشد. ازآنجاکه الگوریتم این روش یک الگوریتم ترتیبی است، بنابراین الگوریتم‌های جایگزینی برای حل معادلات سه‌قطری با رویکرد پردازش موازی در پردازنده‌های گرافیکی نظیر کاهش متناوب (Cyclic Reduction) و کاهش متناوب موازی (Parallel Cyclic Reduction) معرفی‌شده‌اند. این الگوریتم‌ها و نظایر آنها که به همین منظور ابداع‌شده‌اند (مانند الگوریتم‌های موجود در مرجع [2]) ابزارهای مناسبی برای بسیاری از محققان هستند که به‌دنبال تقویت عملکرد کاربرد پردازنده‌های گرافیکی در حل مسائل مختلف می‌باشند.

ژانگ و همکاران [2] در سال ۲۰۰۹ ابتدا روی قابلیت اجرای الگوریتم‌های کاهش متناوب و کاهش متناوب موازی روی پردازنده‌های گرافیکی بحث کرده‌اند. سپس یک روش ترکیبی از الگوریتم‌های معرفی‌شده را پیشنهاد و عملکرد هر الگوریتم را روی حل ماتریس‌های سه‌قطری آزمایش کردند. لازم به توضیح است که الگوریتم کاهش متناوب مشکل تداخل حافظه (Bank Conflict) در حافظهٔ مشترک کارت گرافیکی را دارد و هم‌چنین برای به‌کار بردن الگوریتم‌های کاهش متناوب و کاهش متناوب موازی اندازهٔ دستگاه سه‌قطری حتماً باید محدود به 2^n باشد. در سال ۲۰۱۱ گودک و استرودکا [3] در مورد محدودیت ابعاد ماتریس در حافظهٔ مشترک کارت گرافیکی بحث کرده‌اند. آنها نشان دادند که انتقال پیاپی اطلاعات بین حافظهٔ مشترک و حافظهٔ سراسری کارت گرافیکی بر عملکرد الگوریتم تأثیر می‌گذارد. در سال ۲۰۱۱ دیویدسون و همکاران [4] یک الگوریتم چندمرحله‌ای برای حل دستگاه معادلات سه‌قطری با پردازندهٔ گرافیکی پیشنهاد کردند که نتایج کار آنها نشان می‌داد سرعت الگوریتم پیشنهادی تا پنج برابر افزایش سرعت در مقابل الگوریتم‌های حل غیرموازی دستگاه معادلات سه‌قطری به‌دست آورده است. اگلوف [5] الگوریتم کاهش متناوب موازی را برای حل دستگاه معادلات سه قطری با اندازهٔ بسیار بزرگ در معادلات دیفرانسیل با مشتقات جزئی به‌کار برد و به‌دلیل استفاده از حافظهٔ سراسری، ۶۰ درصد کاهش سرعت را گزارش داد. کیم و همکاران [6] به‌دلیل محدودیت در حافظهٔ مشترک، پیشنهاد دادند که برای دستگاه‌های بزرگ ابتدا آنها را توسط الگوریتم کاهش متناوب موازی به دستگاه‌های کوچک‌تر تبدیل کنند، سپس این دستگاه معادلات مجزا توسط الگوریتم توماس حل شوند. توتکن و همکاران [7] به بحث و بررسی روی

شبیه‌سازی جریان توسط معادلات مرتبه بالا و حل آن به‌کمک پردازنده‌های گرافیکی پرداختند. آنها نشان دادند که عملکرد مؤثر یک الگوریتم پردازش موازی برای شبیه‌سازی عددی مرتبه بالا ازنظر سرعت حل، بستگی زیادی به سرعت حلگر دستگاه سه‌قطری دارد. در سال ۲۰۱۲ اصفهانیان و همکاران [8] با پیاده‌سازی روش‌های مرتبه‌بالا کارایی پردازندهٔ گرافیکی را برای حل معادلات هذلولوی، بررسی کردند. اصفهانیان و همکاران [9] در سال ۲۰۱۳ با به‌کار بستن الگوریتم کاهش متناوب بهینه‌شده به حل و شبیه‌سازی جریان روی استوانه و ایرفویل در دو و سه بعد پرداختند و به افزایش سرعت حل ۱/۹ تا ۱۵/۲ برابر در دو بعد و ۶/۴ تا ۲۰/۳ برابر در سه بعد دست یافتند. در سال ۲۰۱۴ محمودی داریان و همکاران [10] نیز به بررسی و حل معادلات اویلر با کمک پردازندهٔ گرافیکی پرداختند، این ارزیابی توسط دو نوع پردازندهٔ گرافیکی مختلف انجام شد و به حداکثر افزایش سرعت ۱۰۵ برابری نسبت به پردازندهٔ مرکزی دست یافتند.

در این مقاله باتوجه به محدودیت الگوریتم‌های معرفی‌شده در مورد اندازهٔ دستگاه سه‌قطری (حتماً اندازهٔ دستگاه باید 2^n باشد) از روشی به‌نام رهیافت توماس موازی برای حل این دسته از دستگاه معادلات استفاده شده است. این رهیافت که در قالب یک گزارش کوتاه نمایشی-اسلایدنما (Powerpoint presentation) و بدون ارائهٔ جزئیات توسط ساخارنیخ (Sakharnykh) و همکارانش در سال ۲۰۰۹ در پایگاه اینترنتی شرکت انویدیا (Nvidia) قرار داده شده است، چندین دستگاه سه‌قطری را با الگوریتم توماس به‌طور موازی و مستقل از هم حل می‌کند. برای مقایسهٔ سرعت و دقت حل این الگوریتم‌ها مسئلهٔ جریان درون حفره، توسط روش تفاضل محدود، شبیه‌سازی شده است. معادلات حاکم گسسته‌شده و این معادلات به شکل سه‌قطری درآمده و سپس با الگوریتم توماس کلاسیک روی پردازندهٔ مرکزی حل شده‌اند. پس از آن الگوریتم‌های کاهش متناوب، کاهش متناوب موازی و رهیافت توماس موازی برای حل استفاده شده‌اند. نتایج حاصل از حل مسئله جریان درون حفره توسط الگوریتم‌های موازی با نتایج مرجع [11] مقایسه شده و از صحت نتایج اطمینان حاصل شده است. درنهایت زمان حل تمام روش‌های حل اندازه‌گیری و با هم مقایسه شده‌اند. حداکثر افزایش سرعتی که الگوریتم‌های کاهش متناوب، کاهش متناوب موازی و رهیافت توماس موازی اجراشده در پردازندهٔ گرافیکی در برابر الگوریتم توماس کلاسیک از خود نشان می‌دهند به‌ترتیب ۴/۴، ۵/۲ و ۳۸/۴۵ می‌باشد. هم‌چنین در این تحقیق، اثر دسترسی هم‌مکان و غیر هم‌مکان به حافظهٔ سراسری مورد بررسی قرار گرفته و نشان داده شده است که با دسترسی هم‌مکان، افزایش سرعت قابل توجهی(حدوداً دو برابر) برای پردازندهٔ گرافیکی حاصل می‌شود.

پردازش موازی در پردازنده‌های گرافیکی

به‌دلیل موازی بودن ذاتی معماری پردازنده‌های گرافیکی، در سال ۲۰۰۶ سازندگان این تراشه‌ها این ایده را ارائه دادند که اگر پردازنده‌های گرافیکی برای انجام محاسبات پیچیده وضوح تصویر توانا است، دلیل و مانعی وجود ندارد که در محاسبات عددی و علمی دیگر نیز کاربرد نداشته باشند. این تفکر باعث ایجاد مفهوم پردازنده‌های گرافیکی همه‌منظوره شد. امروزه رویکردهای مختلفی برای استفاده از پردازنده‌های گرافیکی همه‌منظوره تعریف شده است که هر یک به ابزارهایی برای ارتباط کاربر و پردازنده (از جمله میان‌افزار کودا (CUDA) و اپن‌سی‌ال (OPENCL) و غیره) نیاز دارد. در واقع این ابزارها چهارچوبی هستند که راه استفاده از پردازنده‌های گرافیکی را برای محاسباتی که ارتباطی با نمایش گرافیکی ندارند هموار می‌سازند. در حال حاضر میان‌افزار کودا پرکاربردترین

ابزار برنامه‌نویسی بر پایهٔ پردازندهٔ گرافیکی است. کودا یک محیط نرم‌افزاری برای محاسبات موازی و مقیاس‌پذیر است که بر پایهٔ زبان ++C/C طراحی شده است. این میان‌افزار مختص پردازنده‌های ساخته شده توسط شرکت انویدیا (Nvidia) می‌باشد.

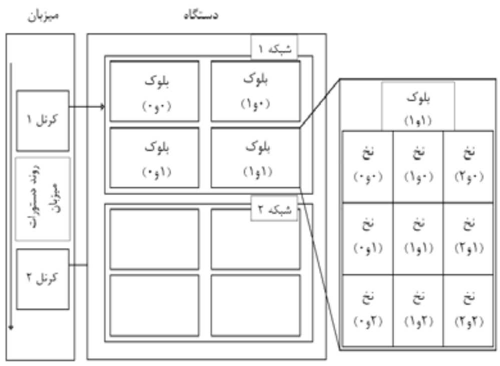
معماری یک پردازندهٔ گرافیکی با یک پردازندهٔ مرکزی تفاوت دارد. پردازندهٔ گرافیکی ترانزیستورهای بیشتری نسبت به پردازنده‌های مرکزی دارد که این ترانزیستورها مسئول محاسبات هستند، اما در مورد واحدهای کنترل جریان برنامه و واحد منطق و هم‌چنین از نظر حافظهٔ درونی نسبت به پردازنده‌های مرکزی ضعیف‌تر می‌باشند. پردازنده‌های گرافیکی، درواقع پردازشگرهای موازی هستند که از سامانه‌های چندنخی (Multi Threads) عظیمی استفاده می‌کنند. برای تحقق این امر هر پردازندهٔ گرافیکی دارای تعدادی چندپردازنده می‌باشد که هر چندپردازنده دارای چندین هستهٔ کودا (CUDA Core) است. هر هسته کودا برای پردازش برنامه و اجرای یک نخ محاسباتی (Computational Thread) اختصاص داده می‌شود. یک نخ محاسباتی یک واحد کوچک نرم‌افزاری و درواقع کوچک‌ترین واحد از دستورات در یک برنامهٔ کامپیوتری است. درمقابل هستهٔ کودا یک واحد کوچک سخت‌افزاری است که ترکیب این دو پردازش موازی را شکل می‌دهد. مجموعه دستورات اجرایی در پردازنده‌های گرافیکی به نام کرنل (Kernel) شناخته می‌شوند که همانند توابع در زبان‌های معمول برنامه‌نویسی هستند با این تفاوت که کرنل‌ها وقتی اجرا می‌شوند هر دستور را یک نخ محاسباتی به‌صورت جداگانه (موازی) اجرا می‌کند. در ساختار پردازش موازی با پردازنده‌های گرافیکی، نخ‌های محاسباتی در واحدهایی به نام بلوک (Block) دسته‌بندی می‌شوند و درنهایت آخرین واحد به نام شبکه که خود دربرگیرنده تعداد زیادی بلوک می‌باشد. این ساختار در شکل (۱) مشخص است. در پردازش موازی باکمک پردازنده‌های گرافیکی به پردازندهٔ مرکزی میزبان (Host) و به پردازندهٔ گرافیکی دستگاه (Device) می‌گویند در شکل (۲) ساختار حافظهٔ دستگاه و میزبان مشخص هستند. هر نخ محاسباتی حافظهٔ ثباتی (Register) دارد که فقط برای همان نخ محاسباتی قابل دسترس است اما حافظهٔ مشترک (Shared-Memory) برای تمام نخ‌های داخل یک بلوک قابل دسترس هستند. حافظهٔ سراسری و ثابت هم توسط نخ‌های تمام بلوک‌ها قابل‌دسترسی هستند. میزبان هم می‌تواند داده‌ها را از این حافظه‌ها دریافت یا به آن‌ها ارسال کند. میزبان نیز حافظه مجزایی دارد که نخها به آن دسترسی مستقیم ندارند.

اصولاً برنامه‌نویسی باکمک پردازنده‌های گرافیکی به این صورت است که دستورات توسط پردازندهٔ مرکزی اجرا می‌شوند و در قسمت‌هایی که به قدرت پردازشی بالایی احتیاج است، توسط کرنل‌ها پردازندهٔ گرافیکی وارد جریان محاسبات می‌شود، پس از اجرای دستورات دوباره پردازندهٔ مرکزی ادامهٔ دستورات را اجرا می‌کند و این روند تا پایان برنامه ادامه پیدا می‌کند. به این رویکرد، در اصطلاح محاسبات ناهمگن (Heterogeneous Computations) می‌گویند (شکل ۱). در پردازش موازی توسط پردازنده‌های گرافیکی علاوه بر قدرت و مشخصات فنی پردازنده‌های گرافیکی، عوامل مختلفی بر افزایش بازده پردازش و کاهش زمان حل مؤثر هستند؛ ازجمله دسترسی هم‌مکان به حافظهٔ سراسری (Coalesced Memory Access)، معطل نماندن نخ‌های محاسباتی در طول پردازش و درگیر بودن آنها و استفادهٔ درست از حافظهٔ مشترک که در بخش‌های بعدی به تفصیل به آنها پرداخته خواهد شد.

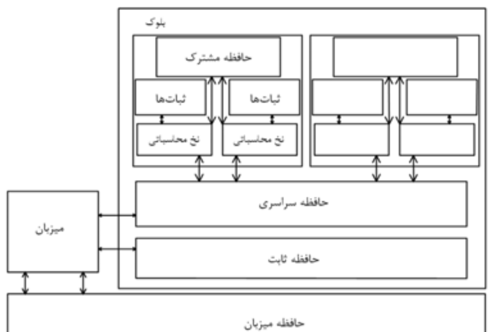
روش‌های حل دستگاه معادلات سه‌قطری

یک حلگر دستگاه معادلات سه‌قطری، الگوریتمی است که بردار مجهولات x را در $Ax = d$ می‌یابد، که A ماتریس ضرایب با سه قطر غیرصفر و d برداری با

مقادیر معلوم است. در ماتریس ضرایب نیز سه قطر اصلی، قطر بالا و قطر پایین به‌ترتیب با b، c و a نمایش داده شده‌اند. الگوریتم‌های متفاوتی برای حل چنین دستگاه معادلاتی وجود دارد اما به این دلیل که الگوریتم توماس یک الگوریتم بسیار سریع است برای به چالش کشیدن توانایی و سرعت پردازنده‌های گرافیکی در حل مسائل، انتخاب شده است. در این تحقیق الگوریتم توماس رایج و شناخته شده به نام الگوریتم توماس کلاسیک ذکر می‌شود تا از رهیافت توماس با رویکرد موازی که در ادامه معرفی می‌شود تمایز داده شود.



شکل ۱ ساختار پردازش موازی در روند اجرای برنامه



شکل ۲ ساختار حافظه در پردازندهٔ گرافیکی

الگوریتم توماس کلاسیک. یکی از سریع‌ترین و پرکاربردترین الگوریتم‌های حل دستگاه معادلات الگوریتم توماس است. الگوریتم توماس شامل دو مرحله است: الف) کاهش پیش‌رو توسط معادلات (۱ و ۲)، ب) جایگزینی پس‌رو یعنی معادلهٔ (۳).

$$c_1' = \frac{c_1}{b_1}\,, c_i' = \frac{c_i}{b_i - c_{i-1}' a_i} \quad i = 2,3,\dots,n-1 \qquad (۱)$$

$$d_1' = \frac{d_1}{b_1}\,, d_i' = \frac{d_i - d_{i-1}' a_i}{b_i - c_i' a_i} \quad i = 2,3,\dots,n \qquad (۲)$$

$$x_n = d_n'\,, x_i = d_i' - c_i' x_{i+1} \quad i = n-1, n-2, \dots, 1 \qquad (۳)$$

الگوریتم کاهش متناوب. الگوریتم کاهش متناوب نیز مانند الگوریتم توماس یک الگوریتم دومرحله‌ای است: (الف) کاهش پیش‌رو، (ب) جایگزینی پس‌رو [2]. در مرحلهٔ پیش‌رو، الگوریتم اندازهٔ دستگاه را در هر مرحله به نصف کاهش می‌دهد و این روند را ادامه می‌دهد تا جایی که به دو معادلهٔ مستقل برسد. در هر مرحله مقادیر قطر اصلی و قطرهای بالا و پایین و سمت راست معادله طبق معادلات (۶-۴) تغییر می‌کنند. پس از آن‌که بااستفاده از معادلات (۶-۴)، دستگاهی به اندازهٔ ۲ در ۲ حاصل شد، این دستگاه دو معادله‌ای با روش‌های مرسوم حل می‌شود. درادامه بااستفاده از معادلهٔ (۷) می‌توان تمام مجهولات دیگر را به‌دست آورد. این مرحله جایگزینی پس‌رو نام دارد.

$$a_i' = -a_{i-1} k_1\,, b_i' = b_i - c_{i-1} k_1 - a_{i+1} k_2 \qquad (۴)$$

$$c_i' = -c_{i+1} k_2\,, d_i' = d_i - d_{i-1} k_1 - d_{i+1} k_2 \qquad (۵)$$

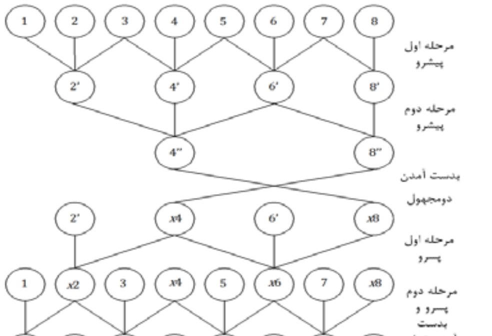
$$k_1 = \frac{a_i}{b_{i-1}}\,, k_2 = \frac{c_i}{b_{i+1}} \qquad (۶)$$

$$x_i = \frac{d_i' - a_i' x_{i-1} - c_i' x_{i+1}}{b_i'} \qquad (۷)$$

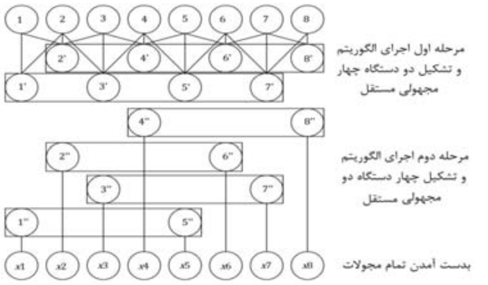
شکل (۳) نشان دهندهٔ الگوی الگوریتم کاهش متناوب برای یک دستگاه به‌اندازهٔ ۸ در ۸ است. با سه بار اجرای الگوریتم روی دستگاه معادلات، با نصف شدن اندازهٔ دستگاه در هر مرحله، در مرحلهٔ سوم دو معادله با دو مجهول به‌دست می‌آید که به‌سادگی قابل حل است.

الگوریتم کاهش متناوب موازی. الگوریتم کاهش متناوب موازی، یک الگوریتم بهبودیافته بر پایهٔ الگوریتم کاهش متناوب است. اما برخلاف الگوریتم کاهش متناوب فقط دارای یک مرحله است و آن مرحله کاهش پیش‌رو است. تفاوت دیگر این دو الگوریتم این است که در الگوریتم کاهش متناوب موازی در هر مرحله ضرایب تمام معادلات تغییر می‌کنند. برای مثال در شکل (۴) مراحل الگوریتم کاهش متناوب موازی برای یک دستگاه ۸ معادله ۸ مجهول به‌صورت شماتیک نشان داده شده‌اند. در هر مرحله از الگوریتم کاهش متناوب موازی دستگاه معادلات به دو دستگاه کاملاً مستقل تبدیل می‌شوند. مرحلهٔ پیش‌رو در الگوریتم کاهش متناوب موازی بر پایهٔ الگوریتم کاهش متناوب است و از معادلات (۶-۴) پیروی می‌کند. اما به‌جای مرحلهٔ پس‌رو که به‌دست آوردن مجهولات بااستفاده از معادلهٔ (۷) است، الگوریتم کاهش متناوب موازی به تعداد مشخصی دستگاه دو معادله دو مجهول می‌رسد که به‌صورت موازی این دستگاه‌های مستقل قابل حل هستند. یکی از برتری‌های الگوریتم کاهش متناوب موازی بر الگوریتم کاهش متناوب در همین امر است که در هر لحظه تمام نخ‌های محاسباتی در حال انجام وظایف خود هستند، اما در الگوریتم کاهش متناوب در بیشتر مراحل، تعدادی از نخ‌های محاسباتی معطل می‌مانند و همین موضوع در عملکرد این الگوریتم تأثیر منفی دارد. بهترین حالت در پردازش موازی این است که در اجرای برنامه تمام نخ‌ها درگیر محاسبات باشند.

رهیافت توماس موازی. رهیافت توماس موازی از این موضوع نشئت می‌گیرد که در مسائل با بیش از یک‌بعد، هر سطر از شبکهٔ عددی (باتوجه به معادلات ۱۲ و ۱۳) می‌تواند یک دستگاه سه‌قطری تشکیل دهد که هر سطر یا هر دستگاه مستقل از دستگاه‌های دیگر می‌توانند حل شوند. در این روش وظیفهٔ حل هر سطر یا هر دستگاه معادلهٔ سه‌قطری به یک نخ محاسباتی سپرده می‌شود. همان‌طور که در شکل (۵) مشخص است هر نخ محاسباتی می‌تواند یک دستگاه معادله را توسط الگوریتم توماس کلاسیک و مستقل از دستگاه‌های دیگر حل نماید. هر نخ محاسباتی بااستفاده از معادلات (۳-۱) می‌تواند دستگاه مربوط به خود را حل نماید. مزیت این روش این است که از سرعت الگوریتم توماس بهره می‌برد و این بهره زمانی که چندین نخ به‌طور هم‌زمان شروع به حل با این الگوریتم می‌کنند، چند برابر می‌شود.



شکل ۳ الگوریتم کاهش متناوب برای یک دستگاه ۸ معادله و ۸ مجهول



شکل ۴ مراحل الگوریتم کاهش متناوب موازی برای یک دستگاه ۸ مجهولی

هندسهٔ مسئله و معادلات حاکم

برای به‌کار بردن الگوریتم‌های معرفی‌شده و بررسی عملکرد، دقت و سرعت این الگوریتم‌ها درمقایسه با الگوریتم توماس کلاسیک، مسئله جریان درون حفره درنظر گرفته شده است. شبیه‌سازی جریان درون حفره یکی از رایج‌ترین مسائل در دینامیک سیالات محاسباتی است، و تاکنون محققان بسیاری در مورد تمام جوانب این مسئله بحث و گفتگو کرده‌اند، و نتایج این تحقیقات منتشر شده‌اند [11-14]. این مسئله به‌علت هندسه و شرایط مرزی ساده‌ای که دارد، معمولاً در روش‌های جدید عددی از نتایج آن به‌عنوان یک نمونه اعتبارسنجی استفاده می‌کنند. در این مقاله، مسئلهٔ جریان درون حفره برای رینولدزهای ۱۰۰، ۴۰۰ و ۱۰۰۰ و در حالت پایدار بررسی و نتایج با مرجع [11] مقایسه شده‌اند. برای شبیه‌سازی جریان درون حفره، روش تفاضل محدود روی معادلات تابع جریان- تاوایی اعمال شده است. معادلات تابع جریان- تاوایی برمبنای تعریف تاوایی و تابع جریان به‌دست می‌آیند. تابع جریان توسط معادلهٔ (۸) و تاوایی نیز توسط معادلهٔ (۹) تعریف می‌شوند. هم‌چنین معادلهٔ (۱۰) معادلهٔ جابه‌جایی- پخشی برای تاوایی است.

$$\boldsymbol{u} = \frac{\partial \psi}{\partial y} \quad , \quad \boldsymbol{v} = -\frac{\partial \psi}{\partial x} \tag{۸}$$

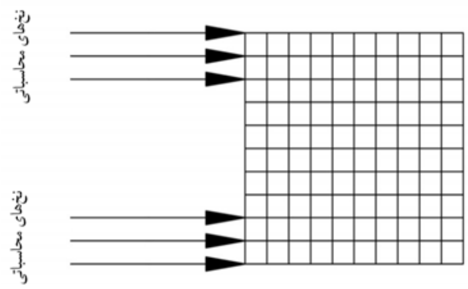
$$\boldsymbol{\omega} = \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}} - \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{y}} \tag{۹}$$

$$\mathbf{u}\frac{\partial \boldsymbol{\omega}}{\partial \mathbf{x}} + \mathbf{v}\frac{\partial \boldsymbol{\omega}}{\partial \mathbf{y}} = \boldsymbol{\nu}\left(\frac{\partial^2 \boldsymbol{\omega}}{\partial \mathbf{x}^2} + \frac{\partial^2 \boldsymbol{\omega}}{\partial \mathbf{y}^2}\right) \tag{۱۰}$$

با قرار دادن معادلهٔ (۸) در معادلهٔ (۹)، رابطه‌ای بین تابع جریان و تاوایی به‌دست می‌آید:

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = -\omega \tag{۱۱}$$

بااستفاده از معادلات (۱۰ و ۱۱) می‌توان جریان لزج تراکم‌ناپذیر دوبعدی را در یک هندسهٔ مستطیلی بدون نیاز به حل معادلهٔ پیوستگی تحلیل کرد. در این مسئله به‌سبب جریان بدون لغزش روی دیواره‌ها، مؤلفه‌های سرعت روی دیواره‌های ثابت صفر است. بنابراین باتوجه به معادلهٔ (۸) می‌توان نتیجه گرفت که تابع جریان روی این دیواره‌ها یک عدد ثابت است (مثلاً $\psi = 0$).

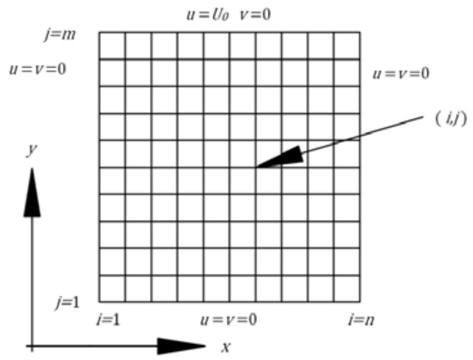


شکل ۵ ساختار نخ‌های محاسباتی برای اجرای رهیافت توماس موازی روی یک شبکهٔ عددی

شرایط مرزی برای تاوایی روی مرزها نیز بااستفاده از معادلهٔ (۱۱) به‌دست می‌آیند. برای حل معادلات، یک شبکهٔ عددی همانند شکل (۶) درنظر گرفته می‌شود و معادلات (۱۰ و ۱۱) با روش تفاضل محدود مرکزی گسسته می‌شوند و با مرتب کردن معادلات به‌ترتیب، معادلات (۱۲ و ۱۳) به‌دست می‌آیند که به‌شکل سه‌قطری مرتب شده‌اند. که $\beta = \frac{\Delta x}{\Delta y}$ و i و j اندیس‌های گره‌های شبکهٔ محاسباتی هستند. برای این تحقیق $\beta = 1$ درنظر گرفته شده است.

$$\psi_{i+1,j} + \psi_{i-1,j} - 2(1+\beta^2)\psi_{i,j} = -\Delta x^2 \omega_{i,j} - \beta^2\left(\psi_{i,j+1} + \psi_{i,j-1}\right) \tag{۱۲}$$

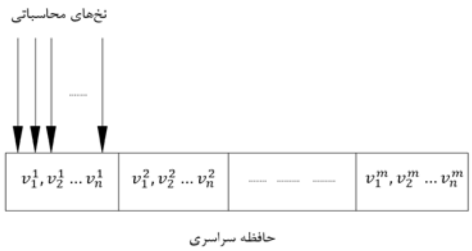
$$\omega_{i+1,j}\left(1+\frac{u_{i,j}\Delta x}{2\nu}\right) + \omega_{i-1,j}\left(1-\frac{u_{i,j}\Delta x}{2\nu}\right) - 2\left(1+\beta^2\right)\omega_{i,j} = \beta^2\left(\omega_{i,j+1}+\omega_{i,j-1}\right) - \frac{\beta v_{i,j}\Delta x}{2\nu}\left(\omega_{i,j+1}-\omega_{i,j-1}\right) \tag{۱۳}$$



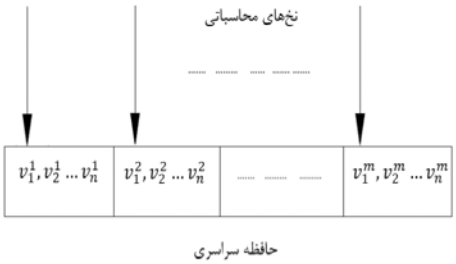
شکل ۶ شبکهٔ عددی برای روش تفاضل محدود
و شرایط مرزی سرعت

پیاده‌سازی الگوریتم روی پردازندهٔ گرافیکی

مسئلهٔ دیگری که در این تحقیق بررسی شده است، ارزیابی تأثیر به‌کار بردن مفهوم دسترسی هم‌مکان به حافظه در زمان حل است. در ساختار سخت‌افزار حافظهٔ دستگاه و میزبان، متغیرها به‌صورت خطی ذخیره می‌شوند. به‌عنوان مثال اگر یک ماتریس در حافظه ذخیره شده باشد، درایهٔ اول از سطر اول ماتریس در مکان مشخصی از حافظه ذخیره می‌شود و به‌ترتیب درایه‌های بعدی در ادامه ذخیره می‌شوند تا آخرین درایهٔ سطر اول، بعد از آن اولین درایهٔ سطر دوم آدرس‌دهی و ذخیره‌سازی می‌شود و به همین ترتیب این روند تکرار می‌شود. برای میزبان به‌علت این‌که ساختار ترتیبی دارد در هر لحظه پردازنده فقط به یک قسمت حافظه نیاز دارد اما در مورد پردازندهٔ گرافیکی به‌علت این‌که چندین پردازنده هم‌زمان در حال پردازش هستند و طبیعتاً هر پردازنده بنا بر دستوری که اجرا می‌کند ممکن است به مقادیر ذخیره شده در قسمت‌های مختلف حافظه نیاز داشته باشد و در واقع فراخوانی هم‌زمان مقادیر از قسمت‌های مختلف حافظه باتوجه به پهنای باند حافظه، می‌تواند سرعت پردازندهٔ گرافیکی را تا حد چشم‌گیری کاهش دهد. برای حل این مشکل در پردازش موازی باید دقت شود که نخ‌های محاسباتی هنگام پردازش به یک منطقه از حافظه نیاز داشته باشند که با یک‌بار فراخوانی از حافظه نیاز تمام نخ‌ها برآورده شود. در شکل‌های (۷ و ۸) به‌ترتیب دسترسی هم‌مکان و دسترسی غیرهم‌مکان (UnCoalesced Memory Access) به مقادیر یک متغیر دوبعدی ذخیره‌شده در حافظهٔ سراسری توسط نخ‌های محاسباتی نشان داده شده‌اند. در شکل‌های (۷ و ۸) اندیس پایین‌نویس بیانگر شمارهٔ ستون و اندیس بالانویس بیانگر شمارهٔ سطر است.



شکل ۷ دسترسی هم‌مکان نخ‌های محاسباتی به حافظه سراسری



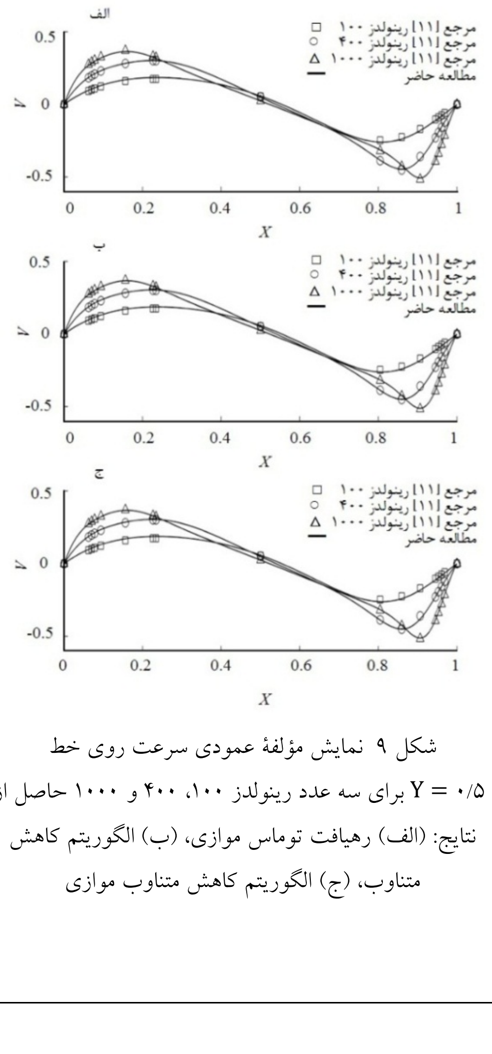
شکل ۸ دسترسی غیرهم‌مکان نخ‌های محاسباتی
به حافظهٔ سراسری

بحث نتایج

در این قسمت ابتدا به بررسی صحیح بودن نتایج حل عددی و مقایسه با نتایج مرجع [11] پرداخته شده است. در این قسمت تمام نتایج برای تطابق با مرجع [11] بی‌بعد گزارش شده‌اند، که عبارتند از $V=\left(\frac{v}{U_0}\right)$، $U=\left(\frac{u}{U_0}\right)$، $X=\left(\frac{x}{L}\right)$، $Y=\left(\frac{y}{L}\right)$ و $\Omega=\left(\frac{\omega L}{U_0}\right)$. در شکل (۹) مؤلفهٔ عمودی سرعت روی خط $Y=$۰/۵ حاصل از حل مسئله توسط رهیافت توماس و دو الگوریتم کاهش متناوب و کاهش متناوب موازی گزارش و با مرجع [11] مقایسه شده است. به همین ترتیب مؤلفهٔ افقی سرعت نیز روی خط $X=$۰/۵ در شکل (۱۰) مورد مقایسه قرار گرفته‌اند. در شکل‌های

(۹ و ۱۰) مقادیر برای نتایج حل با یک شبکه ۱۲۸×۱۲۸ ارائه شده‌اند.

در شکل (۱۱) افزایش سرعت زمان حل مسئلهٔ جریان درون حفره توسط اجرای رهیافت توماس، الگوریتم کاهش متناوب و کاهش متناوب موازی روی پردازندهٔ گرافیکی با زمان حل مسئله توسط پردازندهٔ مرکزی و اجرای الگوریتم توماس کلاسیک برای ۱۰۰۰ تکرار در رینولدز ۱۰۰۰ مقایسه شده‌اند. در شکل (۱۲) افزایش سرعت رهیافت توماس موازی اجراشده روی پردازندهٔ گرافیکی، با دسترسی‌های هم‌مکان و غیرهم‌مکان نسبت به الگوریتم توماس کلاسیک اجراشده روی پردازندهٔ مرکزی، مقایسه شده‌اند. مشخص است که در هر تعداد گره محاسباتی سرعت الگوریتم با دسترسی هم‌مکان حدوداً دو برابر سرعت الگوریتم با دسترسی غیرهم‌مکان است.

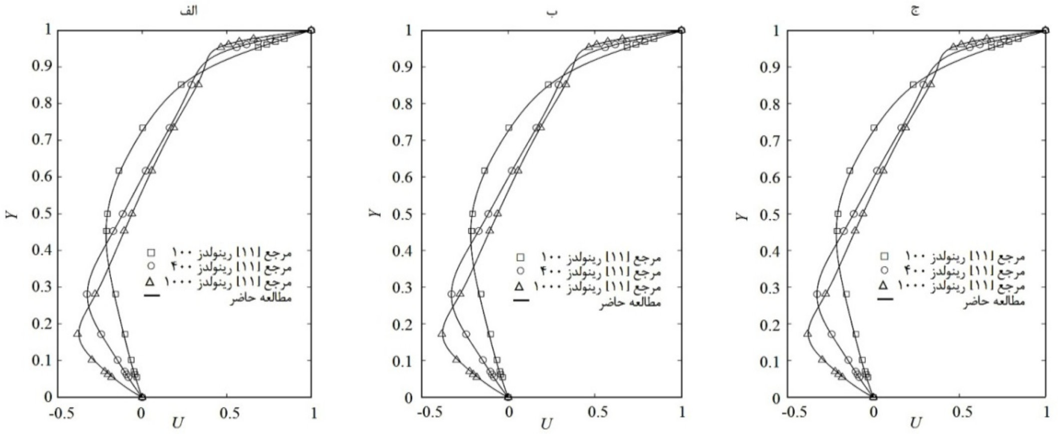


شکل ۹ نمایش مؤلفهٔ عمودی سرعت روی خط $Y = ۰/۵$ برای سه عدد رینولدز ۱۰۰، ۴۰۰ و ۱۰۰۰ حاصل از نتایج: (الف) رهیافت توماس موازی، (ب) الگوریتم کاهش متناوب، (ج) الگوریتم کاهش متناوب موازی

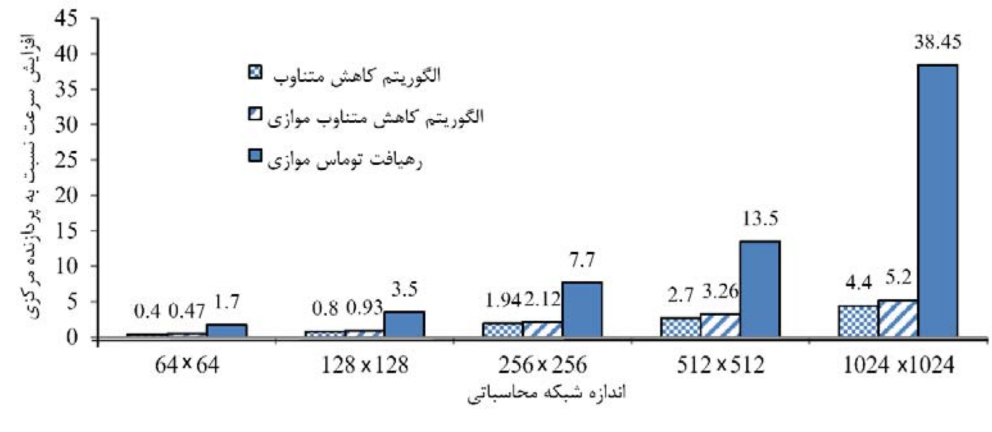
نتیجه‌گیری

در این تحقیق دو الگوریتم کاهش متناوب و کاهش متناوب موازی برای اجرا روی پردازنده‌های گرافیکی معرفی شدند و همچنین برپایهٔ الگوریتم توماس کلاسیک، این ایده مطرح شد که می‌توان دستگاه معادلات مربوط به هر سطر محاسباتی را مستقل از دیگر دستگاه‌ها حل کرد و این نکته منجر به استفادهٔ رهیافت توماس موازی شد. با پیاده‌سازی این الگوریتم‌ها روی پردازندهٔ گرافیکی، اجرا و حل مسئلهٔ جریان درون حفره، نتایج حاصل از حل توسط پردازندهٔ گرافیکی با نتایج حل روی پردازندهٔ مرکزی و مرجع [11] از تطابق قابل قبولی برخوردار بودند. افزایش سرعت زمان حل برای اندازه‌های مختلف شبکه از ۶۴×۶۴ تا ۱۰۲۴×۱۰۲۴ نسبت به زمان حل پردازندهٔ مرکزی، برای الگوریتم کاهش متناوب بین ۰/۴ تا ۴/۴ به‌دست آمد و برای الگوریتم کاهش متناوب موازی از ۰/۴۷ تا ۵/۲ متغیر بود. اما رهیافت توماس موازی نیز با دو نوع رویکرد متفاوت از نوع دسترسی به حافظهٔ سراسری مورد بررسی قرار گرفت. در زمانی که دسترسی از نوع هم‌مکان صورت گرفت حداکثر ۳۸/۵ و در دسترسی غیرهم‌مکان تا ۲۰/۳ برابر، افزایش سرعت حاصل می‌شد. در دسترسی هم‌مکان در تمام اندازه‌های شبکهٔ محاسباتی رهیافت توماس موازی در حل مسئلهٔ افزایش سرعت نشان داد اما هنگامی که نوع دسترسی به حافظه بهینه نبود، سرعت حل بالا نبود و حتی در اندازه شبکهٔ ۶۴×۶۴ کاهش سرعت نسبت به زمان حل پردازندهٔ مرکزی ثبت شد. روش اجرای رهیافت توماس موازی به‌گونه‌ای است که برای دیگر روش‌های حل نیز کاربرد خواهد داشت. زمانی که بتوان هر سطر محاسباتی را مستقل از دیگر سطرها حل نمود، محاسبات هر سطر را یک نخ محاسباتی می‌تواند به‌عهده گیرد خواه دستگاه معادلهٔ تشکیل‌شده سه‌قطری باشد، خواه پنج‌قطری؛ یعنی می‌توان معادلات را با

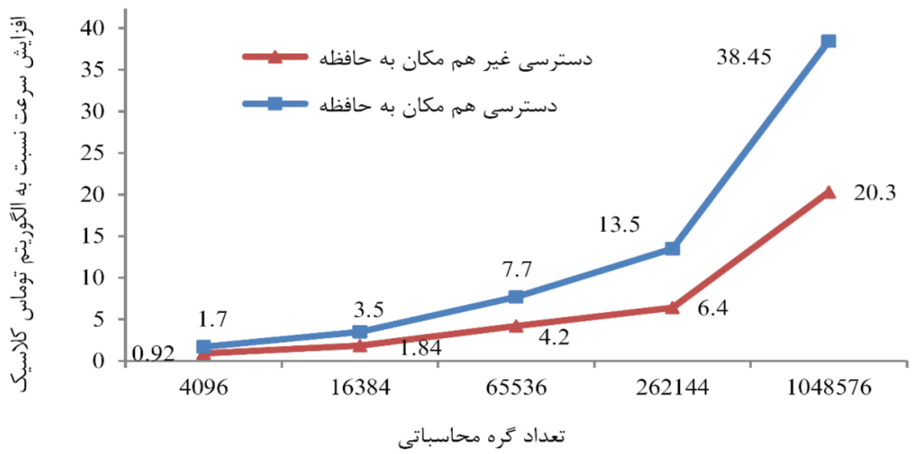
مرتبهٔ بالاتر گسسته کرد (افزایش درگیری تعداد گره‌های محاسباتی) که طبیعتاً ماتریس ضرایب از حالت سه‌قطری خارج می‌شود و برای مثال به یک ماتریس پنج‌قطری تبدیل می‌گردد. در این شرایط می‌توان از روش‌هایی نظیر به‌کار بردن الگوریتم توماس بلوکی برای هر سطر یا مطابق مراجع [15, 16] اعمال روش ضمنی برای متغیر فشرده (ADI Compact) به‌منظور تبدیل معادلات گسستهٔ مرتبه‌بالاتر به یک دستگاه سه‌قطری، استفاده نمود و سپس برای حل آنها از الگوریتم‌های معرفی‌شده و ترجیحاً رهیافت توماس موازی بهره گرفت. از دیگر مزیت‌های رهیافت توماس موازی، محدود نبودن اندازهٔ دستگاه معادلات به ۲^n ذکر شد. این توانایی می‌تواند گسترهٔ کاربرد این رهیافت را افزایش دهد. درنهایت از نتایج ارائه‌شده می‌توان دریافت که استفاده از پردازنده‌های گرافیکی به‌عنوان شتاب‌دهنده به محاسبات عددی، یک ابزار کارآمد و بسیار سریع و مقرون‌به‌صرفه به‌حساب می‌آیند و می‌توان در آینده برای تحقیقات گوناگون از قدرت آنها سود برد.



شکل ۱۰ نمایش مؤلفهٔ افقی سرعت روی خط $X = ۰/۵$ برای سه عدد رینولدز ۱۰۰، ۴۰۰ و ۱۰۰۰ حاصل از نتایج: (الف) رهیافت توماس موازی، (ب) الگوریتم کاهش متناوب، (ج) الگوریتم کاهش متناوب موازی



شکل ۱۱ نمایش افزایش سرعت زمان حل الگوریتم‌های موازی برای رینولدز ۱۰۰۰ در طی ۱۰۰۰ تکرار



شکل ۱۲ مقایسهٔ افزایش سرعت پردازش موازی دربرابر پردازندهٔ مرکزی در حالت دسترسی هم‌مکان و غیرهم‌مکان به حافظه برای تعداد گره محاسباتی مختلف در رینولدز ۱۰۰۰

فهرست علائم

a قطر پایین در ماتریس سه‌قطری
A ماتریس ضرایب
b قطر اصلی در ماتریس سه‌قطری
c قطر بالا در ماتریس سه‌قطری
d بردار مقادیر معلوم
i شمارندهٔ گره محاسباتی در راستای محور طول
j شمارندهٔ گره محاسباتی در راستای محور عرض
L مقدار طول و عرض حفره (m)
m تعداد سطرها
n تعداد ستون‌ها
U_0 سرعت روی مرز متحرک (ms-1)
u مؤلفهٔ افقی سرعت (ms-1)
U مؤلفهٔ افقی سرعت (بدون بعد)
v مؤلفهٔ عمودی سرعت (ms-1)
V مؤلفهٔ عمودی سرعت (بدون بعد)
x محور طول (m)
X محور طول بدون بعد
y محور عرض (m)
Y محور عرض بدون بعد

علایم یونانی

ψ تابع جریان (m2 s-1)
ω تاوایی(s-1)
Ω تاوایی (بدون بعد)
υ لزجت سینماتیکی (m2 s-1)

مراجع

1. Accessed 30 November, 2014; http://www.top500.org/.
2. Zhang, Y. Owns, J.D. and Cohen, J., "Fast Tri-diagonal Solvers on the GPU", *Proceeding of the 15th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, Bangalore, January 09-14, (2010).

3. Göddeke, D. and Strzodka, R., "Cyclic Reduction Tri-diagonal Solvers on GPUs Applied to Mixed-Precision Multigrid", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 22, No. 1, (2011).
4. Davidson, A., Zhang, Y. and Owens, J.D., "An Auto-tuned Method for Solving Large Tri-diagonal Systems on the GPU", *in Proceedings of the 25th IEEE International Parallel and Distributed Processing Symposium*, Anchorage, Alaska, (2011).
5. Egloff, D., "High performance finite difference PDE solvers on GPUs", Quant Alea GmbH, Technical report, February (2010).
6. Kim, H.S., Chang, L.W., Wu, S. and Hwu, W.W., "A Scalable Tri-diagonal Solver for GPUs", *International Conference on parallel processing*, Taipei, pp. 444-453, 13-16 Sept. (2011).
7. Tutkun, B. and Edis, F.O., "A GPU application for high-order compact finite difference scheme", *Computers & Fluids*, Vol. 55, No. 11, pp. 29-35, (2012).
8. Esfahanian, V., Darian, H.M. and Gohari, S.M.I. "Assessment of WENO schemes for numerical simulation of some hyperbolic equations using GPU", *Computers & Fluids*, Vol. 80, pp. 260-268, (2012).
9. Esfahanian, V., Baghapour, B., Torabzadeh, M. and Chizari, H., "An effcient GPU implementation of cyclic reduction solver for high-order compressible viscous flow simulations", *Computers & Fluids*, Vol. 92, pp. 160-171, (2014).
10. Darian, H.M. and Esfahanian, V. "Assessment of WENO schemes for multi-dimensional Euler equations using GPU", *Numerical Methods in Fluids*, Vol. 76, pp. 961-981, (2014).
11. Ghia, U., Ghia, K.N. and Shin, C.T., "High-Re solutions for incompressible flow using the Navier Stokes equations and a multigrid method", *Journal of Computational Physics*, Vol. 48, No. 3, pp. 387-411, (1982).
12. Bicudo, P. and Cardoso, N., "Time dependent simulation of the Driven Lid Cavity at High Reynolds Number", *Physics of Fluid Dynamics*, Vol. 1, pp. 1−20 (2009).
13. Erturk, E., "Discussions on Driven Cavity Flow", *Numerical Methods in Fluids*, Vol. 60, pp. 275-294, 2009.
14. Poochinapan, K., "Numerical Implementations for 2D Lid-Driven Cavity Flow in Stream Function Formulation", *ISRN Applied Mathematics*, Vol. 2012, Article ID 871538, pp. 1-17, (2012).
15. Tiana, Z.F. and Geb, Y.B., "A Fourth-Order Compact ADI Method for Solving Two-Dimensional Unsteady Convection–Diffusion Problems", *Journal of Computational and Applied Mathematics*, Vol. 198, No. 1, pp. 268 – 286, (2007).
16. Erturk, E., "Numerical Performance of Compact Fourth Order Formulation of the Navier-Stokes equations", *Communications in Numerical Methods in Engineerin*, Vol. 24, pp. 2003-2019, (2008).